Localized Triple Modular Redundancy vs. Distributed Triple Modular Redundancy on a ProASIC3E Reprogrammable FPGA

Alex McGuffey*
Valparaiso University, Valparaiso, IN, 46383

Melanie Berg†
MEI, Greenbelt, MD, 20770

Jonathan Pellish‡
NASA, Greenbelt, MD, 20770

Field programmable gate arrays (FPGA) are used in every space application. Currently, most space flight applications use radiation hardened (RH) FPGAs, which are very expensive. There is a desire to use cheaper, commercial off the shelf reprogrammable FPGAs, which are more susceptible to radiation effects known as single-event effects (SEE). The RH parts have SEE and total ionizing dose (TID) hardened elements pre-integrated into the part. This means that the designer does not need to implement any hardening techniques while configuring the device. The COTS parts on the other hand must be mitigated by design in order to insure any form of mitigation. The design techniques this project examines concern the use of localized triple modular redundancy (LTMR) and distributed triple modular redundancy (DTMR). LTMR triples every flip flop in the device architecture while DTMR triples everything except for the global routes (clocks, resets, and enables). The testing was performed on a ProASIC3E FPGA at the Texas A&M cyclotron facility. Two design architectures were used: shift registers and counters, both with LTMR and DTMR mitigation techniques. The test results prove that DTMR is more effective at reducing SEE than LTMR. We also determined that there was not a significant difference between the use of shift registers and counters for test purposes. More testing is required to obtain additional linear energy transfer values for each architecture and mitigation technique in order to determine the most cost-effective method of SEE mitigation.

Nomenclature

σ = cross section (cm^2/bit) - a measure of probability of errors

I. Introduction

Field programmable gate arrays (FPGAs) are critical pieces of space flight hardware, but are very susceptible to radiation damage if not mitigated. The majority of FPGAs in space are radiation hardened (RH) and built with anti-fuse technology which means that once the hardware is configured, a machine fuses the connections together. This process means that the FPGA cannot be reprogrammed ever again. There is a desire to start using reprogrammable FPGA's like the ProASIC3E, which will allow an in-flight reconfiguration. These are also

* Student Intern, NASA Goddard Space Flight Center, Greenbelt, MD 20771
† Senior Engineer, Code 561.4 Radiation Effects and Analysis Group, Building 22 Room 052, NASA Goddard Space Flight Center
‡ Aerospace Engineer, Code 561.4 Radiation Effects and Analyses Group, Building 22 Room 060, NASA Goddard Space Flight Center

commercial off the shelf parts which are much cheaper than their space specific brothers. However, these parts are much more susceptible to the radiation effect known as a single-event effect (SEE).

A SEE occurs when a single ion strikes electronic hardware and causes either a destructive or non-destructive error. Destructive errors are latchups (SEL) where the device gets stuck in a high current state, single-event burnouts (SEB) where the device draws too much current and burns out, and single-event gate ruptures (SEGR) where a gate is destroyed in a power MOSFET.[1] Non-destructive errors are single-event upsets (SEU) where there is a temporary change in a bit or memory, single-event functional interrupts (SEFI) where an upset corrupts a control path and the device temporarily loses functionality[1], and single-event transients (SET) where a voltage or current has a temporary change for a short amount of time.

The radiation environment in space that causes SEEs are proton trapped radiation, solar particles, and galactic cosmic rays. The energy levels of all these particles are measured by the linear energy transfer (LET). To analyze radiation effects, the number of events must first be counted. SEE are counted whenever the output value is not the same as the expected output. For example, if the output is supposed to be 55555 (in hex), but instead is 5x555, where x is any hex value other than 5, then an error is counted. The total number of errors is then divided by the fluence, the number of particles which intersect a unit area, to calculate the cross section which is assigned the Greek letter sigma σ.

In order to evaluate hardware for heavy ion SEE, heavy ions are accelerated in a cyclotron and then slammed into the hardware being tested. Each heavy ion species accelerated to a particular energy has a distinct LET value, which can be modified by changing the angle of incidence or kinetic energy[§]. When angle of incidence is used as an independent variable to modify LET, the metric is then called effective LET. In space, there are generally a larger number of lower LET values than higher ones, and so despite higher LET values having a higher σ, they are usually less frequent.[2] The threshold LET (LET_{th}) is the LET value where the first event is observed. If there is a higher onset threshold, the less likely it is that ions will cause some kind of radiation event.

The probability of radiation errors, $P(Fs)_{error}$, on an FPGA is determined by the errors in the memory as well as the upsets and transients in the logic, on clocks, resets, and enables.[3] There are four parts to this probability: the probability of configuration errors, the probability of D-flip-flop upsets, the probability of dynamic transients that become upsets, and the probability of functional interrupts. Figure 1[3] shows how equation (1) is determined. Since the ProASIC3 FPGA is flash electrically erasable programmable read-only memory (EEPROM) based rather than SRAM based, the probability of configuration errors is diminished. This is because flash is designed so that lowering the voltage will not change to configuration. Since SEE cause a lower voltage, flash is not affected by SEE the same as SRAM is.

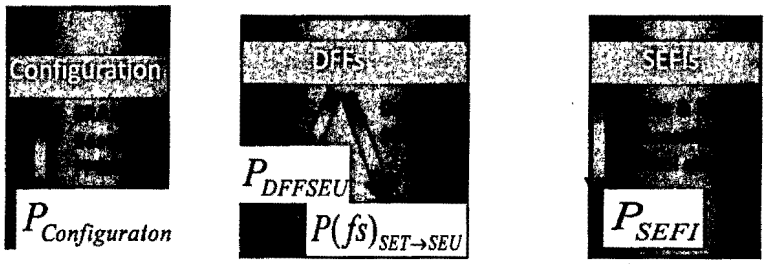

Figure 1. This shows how the probability of errors caused by single event effects in an FPGA is determined.

$$P(fs)_{error} \propto P_{configuration} + P_{DFFSEU} + P(fs)_{SET\rightarrow SEU} + P_{SEFI} \quad (1)$$

To mitigate radiation effects in a FPGA there is a design technique called triple modular redundancy (TMR), which triplicates certain logic on the FPGA and votes on those signals. This way, if one of the signals is corrupted due to a SEE, the logic result would not be adversely affected. There are three main methods of TMR: Localized (LTMR), Distributed (DTMR), and Global (GTMR). LTMR triplicates the flip-flops only, DTMR triplicates everything except the clocks, resets, and enables, and GTMR triplicates everything. This project examined the effectiveness of LTMR versus DTMR within an Actel ProASIC3E flash FPGA.

[§] The angle the hardware is tilted with respect to its normal with the particle

The ProASIC3E FPGA underwent SEE testing at Texas A&M Cyclotron Facility in May 2010 to evaluate susceptibility to SEE. The test engineers configured the chip for multiple tests to check the different effects of the mitigation techniques and to calculate the heavy ion cross section. They first used six different channels of shift registers: two had no logic in between the flip-flops for controls, two had either 8 inverters or 8 buffers, and two had either 20 inverters or 20 buffers between flip-flops. Then counters were used to determine the affect of radiation on fan out signals. Both implementations were designed in LTMR and DTMR, tested at multiple frequencies and linear energy transfer levels using copper and xenon ions, and were exercised with different data patterns.

II. ProASIC3E Architecture and Configuration

The full part number for the ProASIC3E that was tested was A3PE3000- PQG280- 2HJ3T with a lot date code of 0832. It has a total of 3,000,000 system gates. This FPGA consists of six clock conditioning circuits (CCC), 1 kbit of flashROM, 75,264 VersaTiles, 112 4,608-bit blocks, 620 I/O connections, and an ISP AES decryption. The diagram for the FPGA can be seen in Fig 2[4]. Since it is nonvolatile flash based and not SRAM based, the flash memory retains the configuration design when the device is powered off, so that when it is powered back on, the flash will automatically reconfigure the device. The VersaTiles can be used as a 3-input look-up-table (LUT-3) equivalent, a D-flip-flop (DFF) with clear or set, or an enable DFF with a clear or set, are the sequential and combinatorial logic blocks. These possible configurations can be seen in Fig 3.[4]

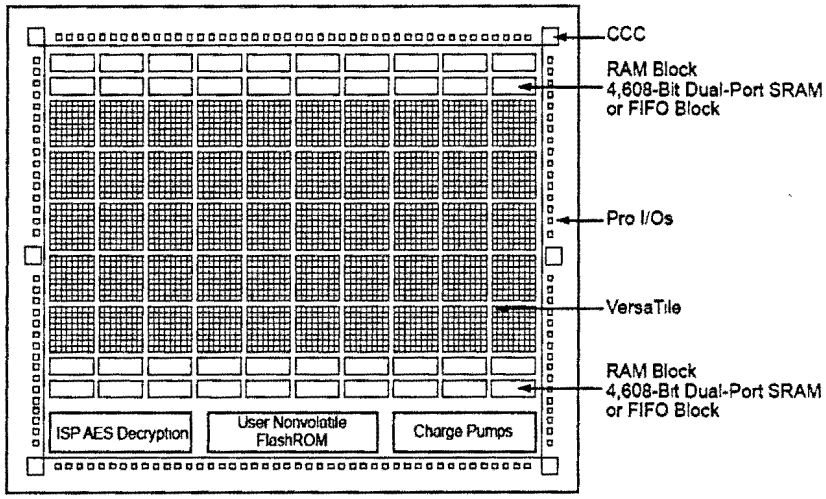

Figure 2. ProASIC3E Device Architecture Overview.

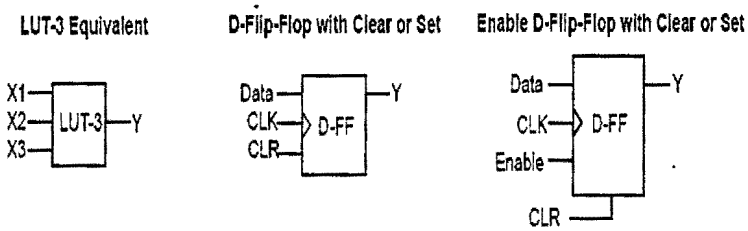

Figure 3. The three posibilites for the VersaTile configuration.

The look up tables can function as any 3-input logic gate. The three inputs act as select lines for transistors which pass one of the references through based upon whether the input was a '0' or a '1'. The references are assigned either a '0' or a '1' as well since this is a digital device, based upon the truth table for the function that is desired by the designer. (Fig 4)

A. Triple Modular Redundancy

When designing the configuration for space flight, the designer must think about mitigation techniques for radiation effects. One of the primary mitigation techniques is called triple modular redundancy (TMR). Originally, TMR was used on the module level and for protecting sequential logic from upsets[5]. As upsets in combinatorial logic are more prevalent than before, TMR may be necessary to mitigate SEE there as well.

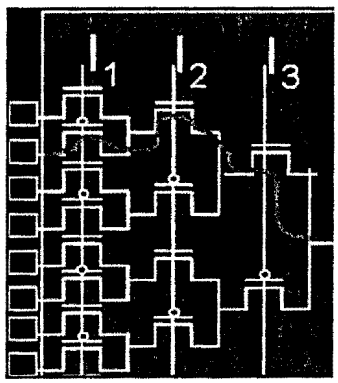

Figure 4. A 3-Input LUT with a path corresponding to input values "011". Notice how the '0' corresponds with the circle to determine the path.

ion

LTMR is the simplest design process of TMR to mitigate SEE. Each DFF is tripled and then sent to a voter before the signal is sent to the next set of combinatorial logic. The triplication of the DFF's helps to prevent the single event upsets in those flip-flops as shown in equation (2). The majority voter then determines which signal to pass onward. The difference between not using TMR and the use of LTMR can be easily seen in Fig 5. The voter for TMR is simply combinatorial logic. It consists of three 2-input AND gates, and a 3-input OR gate. Each flip flop is AND'ed with the others and all three AND gates are then sent to an OR gate. This assures that two of the flip-flop signals must be a '1' for the output to be a '1', or two must be a '0' for the output to also be a '0'.

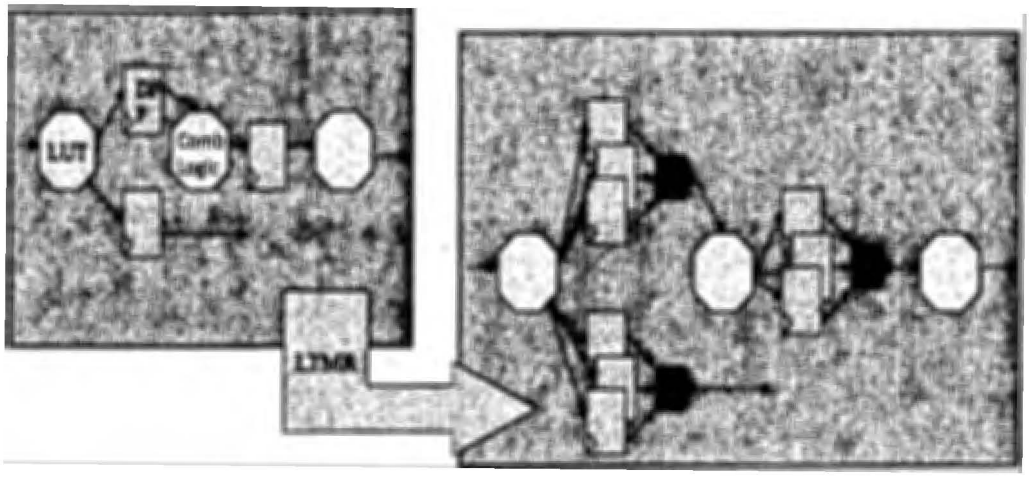

Figure 5. The first image is what the ProASIC3 FPGA configuration might look like with no TMR. The second image shows the LTMR changes.

$$P(fs)_{error} \propto \cancel{P_{configuration}}^{\text{Low}} + \cancel{P_{DFFSEU}}^{\text{Low}} + P(fs)_{SET \rightarrow SEU} + P_{SEFI} \quad (2)$$

The problem with LTMR is that it does nothing to prevent SETs in the combinatorial logic or even the voters. These SETs can be caught by the flip-flops and become SEUs, which get passed to all three flip-flops, completely bypassing the value of the TMR. In order for a transient to be caught by a flip-flop, several things must happen. It must first have sufficient amplitude to affect the system. Then the transient must propagate through the LUT and appear on the output. The DFF the signal is on its way to must be enabled, for any signal to affect it as well. Finally, the transient must occur at the same time as the clock edge in order to be capture by the DFF. This process is shown in Fig 6[3].

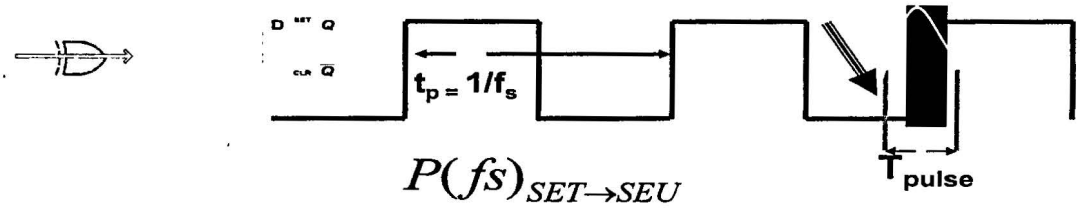

Figure 6. A transient being captured by the DFF can only occur during the clock edge as shown in this diagram.

Since so many things must be perfect for the transient to affect the flip-flops, the probability of SETs leading to SEU is not as high as SEU events occurring on their own. However, SETs will still cause errors if they are not mitigated. Since LTMR does not mitigate SETs another method was also examined. This method is called Distributed Triple Modular Redundancy or DTMR. It consists of the triplication of not only the flip-flops, but the combinatorial logic, and the voters as well (Fig 7). DTMR should mitigate SETs and SEUs as shown in equation (3). The downside to DTMR is that it takes more power and space to accomplish the same task. For this reason, cost-effectiveness must be taken into account.

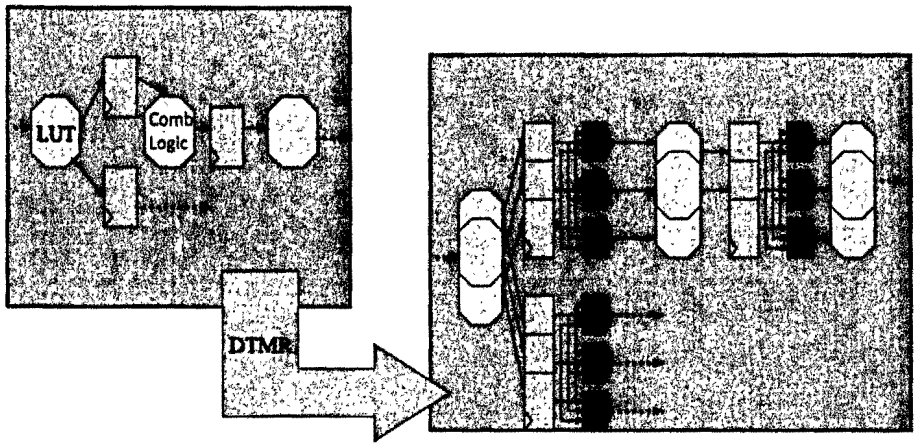

Figure 7. The left image is a diagram of the ProASIC3E design with no mitigation; the right image is a diagram of DTMR mitigation for the same design.

$$P(fs)_{error} \propto P_{configuration}^{\text{Low}} + P_{DFFSEU}^{\text{Low}} + P(fs)_{SET \rightarrow SEU}^{\text{Low}} + P_{SEFI} \quad (3)$$

As equation (3) shows, the probability of SEFI errors is still not diminished. These are caused by errors on the global routes. In order to reduce the SEFI errors, a designer would need to triplicate global routes as well, which is done in the mitigation technique Global TMR. To do so however, would cause many timing difficulties and an even larger power and size requirement.

B. Testing Architecture

For testing purposes, a design architecture had to be established which would enable us to easily determine when and where SEE errors occur. This architecture also must be available to all FPGAs in order to compare these mitigation techniques to other FPGAs. Two architectures were chosen: a shift register architecture and a counter architecture. The shift register architecture has been used in tests before on the RTAX-S, and Xilinx FPGAs[6], so it obvious that the same implementation should be used in the ProASIC3E. The counter architecture was chosen in order to determine if it is a worse case for SEE than the shift register. It is critical to use a design which is complex enough to simulate a space flight application, but also must be as susceptible to SEE as that space flight application.

Each flip-flop in the shift register architecture was linked to the same clock signal so that all the registers would be synchronized. The FPGA had six channels which were used for slightly different versions of the shift register architecture. Each channel had a specific number of either inverters of buffers in between each flip-flop. Table I shows the details of each channel. In order to examine the output values of the shift register, a window of an additional four DFFs was placed into the design as shown in Fig 8[6]. These flip-flops would capture the last four bits of the shift register every four cycles. This window would then send those signals to four outputs which were collected as the data. The window was used in order to allow full functionality of the shift register while still allowing us to capture the last four bits.

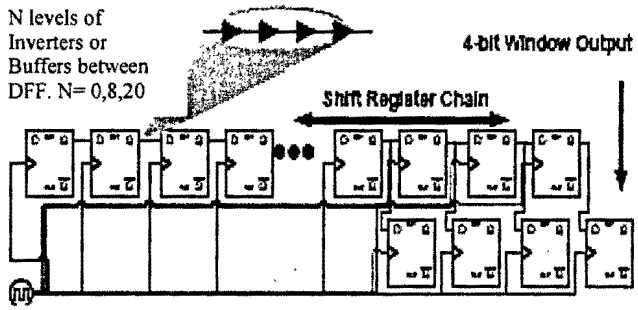

Figure 8. The shift register architecture for the ProASIC3E FPGA testing.

Table 1

Channel	0	1	2	3	4	5
Inverters or Buffers	None	None	Inverters	Buffers	Inverters	Buffers
# Inv/Buff	0	0	8	8	20	20
# DFFs	400	400	280	280	200	200

Table 1. The details regarding the number of inverters, buffers, and D flip-flops within the shift register architecture.

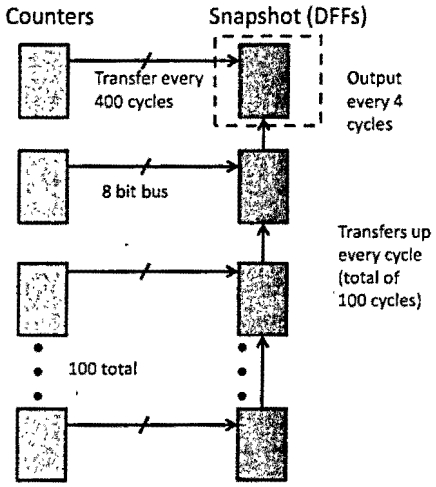

Figure 9. A diagram of the counter-snapshot architecture.

The counter architecture consisted of 100 separate counters, each starting one value away from the others, meaning counters went from 0-99. Each counter was linked to the same clock, but would not affect the other counters in any way. Every 400 clock cycles the counters would pass their values to a system called a snapshot. This snapshot was a shift register with 100 DFFs. Each DFF was linked to a distinct counter so that all the values in the counters would be available in the snapshot. The least significant bit of the shift register was the output from where the data was collected, so every four clock cycles the values would shift one DFF toward the output. This way after 400 cycles, every value which was passed into the snap shot would have been collected as data. The snapshot system allowed the counters to continue counting unhindered, and provided a method to efficiently collect the data. A diagram of the counter-snapshot setup can be seen in Fig 9.

III. Test Methodology

The ProASIC3E FPGA was tested with several different conditions and setups within the architecture already described. Variations included LET values, loaded shift register patterns, and frequencies. The device was tested at Texas A&M University Cyclotron (TAMU) facility on May 24th 2010 by Melanie Berg and Chris Perez, contractors at the NASA Goddard Space Flight Center.

A. Test Conditions and Materials

The ProASIC3E was connected to a testing device called the Low Cost Digital Tester Version 2 (LCDT-V2) via a daughter board. This device was developed by the Radiation Effects and Analysis Group as a re-usable digital device tester. It is built on a Xilinx Spartan 3 FPGA and the daughter board is connected to it with the device under test (DUT). The FPGA controls this daughter board, and is controlled by LabView software by the tester. The LCDT-V2 can be seen in Fig 10, and the diagram of the connection is depicted in Fig 11.

During testing, the LCDT-V2 is placed in front of the cyclotron beam, so that the DUT will be directly struck by the beam. The experiment used beams of copper and xenon ions with energies of 15 MeV/amu. The base LET of the copper was 20 $(MeV \cdot cm^2)/mg$, while the base LET of xenon was 53 $(MeV \cdot cm^2)/mg$. In order to obtain LET values of 75 and 106

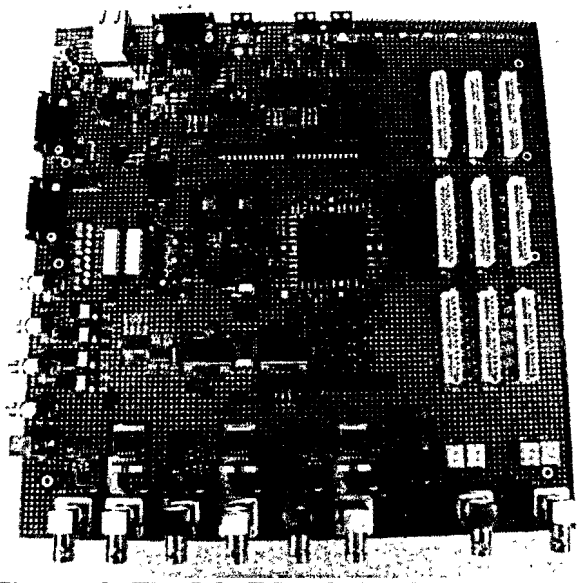

Figure 10. The LCDT-V2 designed by the Radiation Effects and Analysis Group.

(MeV·cm^2)/mg, the incident angle was changed while using the xenon ion. When the DUT was tilted 45° from the normal of the beam, the effective LET became 75 (MeV·cm^2)/mg, and when it was tilted at 60°, the effective LET was 106 (MeV·cm^2)/mg. Figure 12 shows the LCDT-V2 connected with the DUT at the TAMU cyclotron.

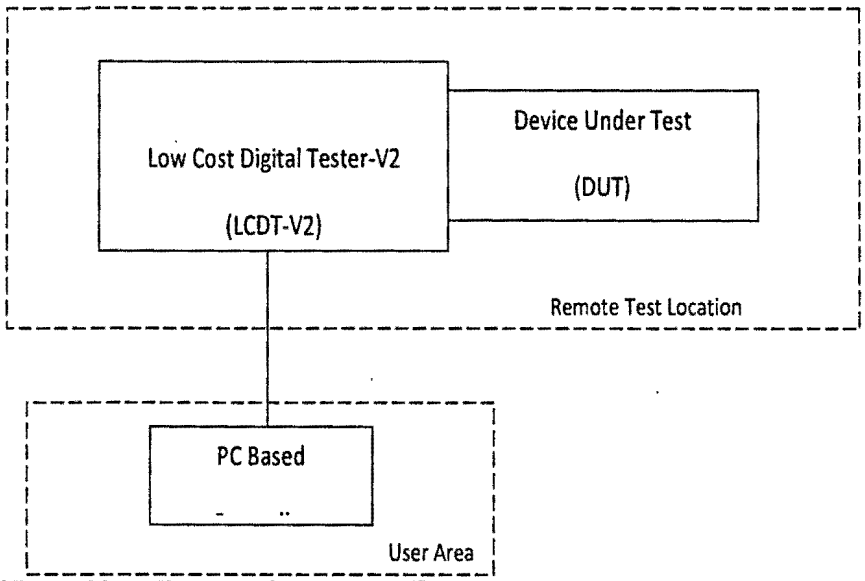

Figure 11. A diagram of the LCDT-V2 connected with the DUT and LabView

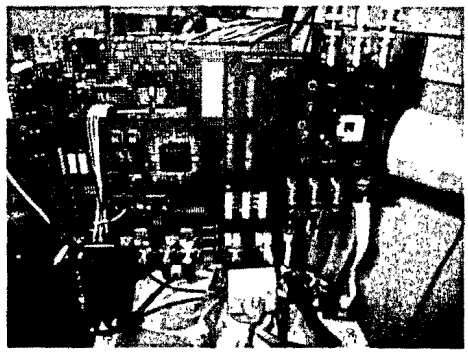

Figure 12. The LCDT-V2 with daughter board and ProASIC3 FPGA as the DUT.

A LabView control panel was designed by Hak Kim for running the ProASIC3 tests. The front panel can be seen in Fig 21 in the index. The tester board and DUT were connected to 3.3V power supplies. The tester was configured first for the type of test being run, followed by the DUT. Using LabView, the test engineer would reset and clear the chip back to its configuration. Then a frequency and pattern were chosen and sent into the device. Finally they would start the test and using another computer with a waveform simulator (fig 22 in the index), they captured the outputs from the channels being tested. Once the test was up and running, the ion beam was turned on and any changes in the output were collected and saved as the data.

There were 44 tests run with different modifications for each test. Out of those 44, only 36 were properly tested, one of which was duplicated. There were 35 different tests run on the FPGA within 5 separate ProASIC3 FPGA DUTs. The modifications already discussed were the LTMR vs. DTMR, the architecture method, and the LET values. Frequencies and shift register patterns were also changed throughout testing. For the LTMR shift register architecture, the tested operating frequencies were 1, 50, and 100 MHz. For the DTMR shift register architecture, the operating frequencies were 1 and 50MHz. The frequencies used on the counters were 8 and 80 MHz.

The shift registers were loaded with one of two patterns, either a checkerboard pattern, or a '0's pattern. The checkerboard pattern consisted of alternating'1's and'0's, which pass through the shift register causing each bit to alternate every clock cycle. The '0's pattern, was merely forcing all the bits to be '0'. These patterns were used because it was easy to identify when there was an error in the shift register. All these different modifications were used as comparison purposes throughout the testing.

IV. Results

To calculate the SEE cross sections for the DUT, the data had to be examined to count the total number of events on each channel of the architecture. The number of events per channel was calculated by the sum of the clock errors per channel and the number of time a bit error occurred per channel. The cross sections were calculated by dividing the total number of events per channel by the fluence on that part. Comparisons were made between the LTMR and DTMR for 50 MHz and 1 MHz for the shift registers, between inverters and buffers used on the shift registers, between checkered patter and '0's pattern for both LTMR and DTMR for the shift registers, between

LTMR and DTMR for the counters and snapshot, and between the counters and shift registers. For all of the following charts, the term in the legend xBxI stands for x number of Buffers and x number of Inverters between DFFs on the shift registers.

A. LTMR vs. DTMR for Shift Registers

The following charts show the cross sections vs. LETs for the LTMR and DTMR data from the shift registers.

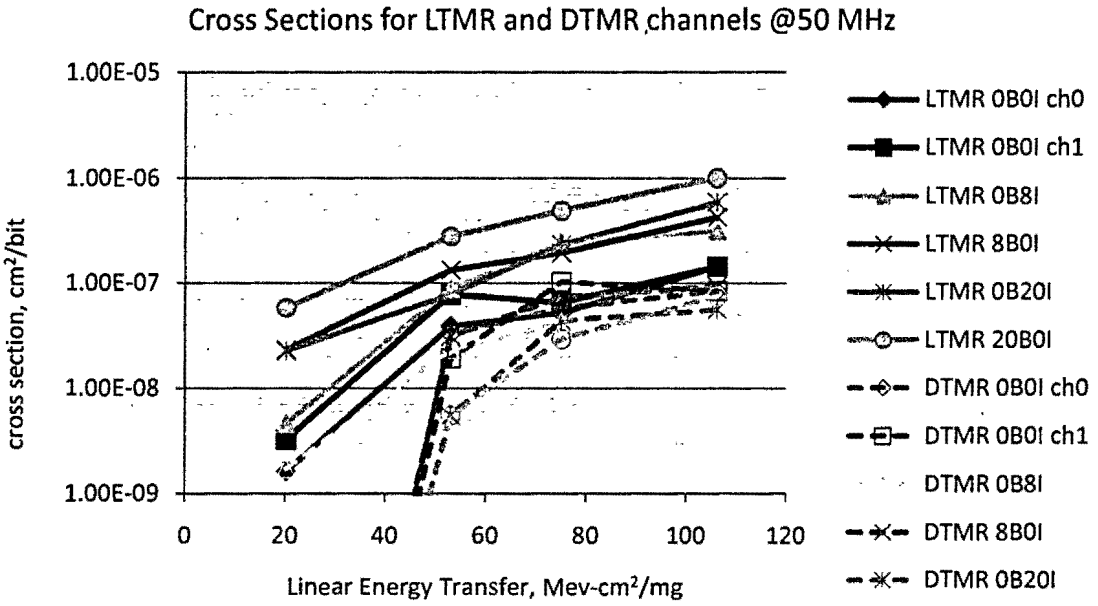

Figure 13. The cross section of the shift registers at 50MHz for LTMR and DTMR. LTMR values are the solid lines, while DTMR values are the dashed lines.

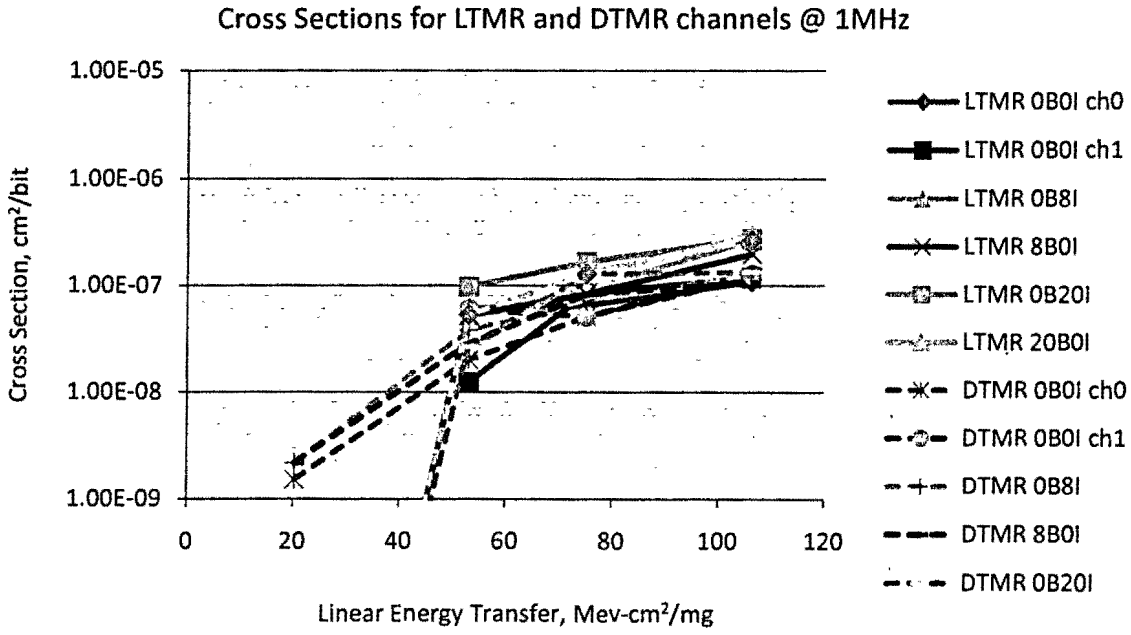

Figure 14. The cross sections of the shift registers at 1MHz for LTMR and DTMR. LTMR values are the solid lines, while DTMR values are the dashed lines.

These two graphs show the relationship between LTMR and DTMR for both 50 MHz and 1 MHz operating frequencies. The important part of the graph is at the lower levels of LET because those determine the LET_{th} which is the point at which the first events are found. By definition, at LET values lower than the LET_{th}, there would be no errors due to single events. In Fig 13, note that the LET_{th} for the DTMR values is between 20 (MeV·cm^2)/mg and 53 (MeV·cm^2)/mg, while the LTMR LET_{th} is lower than 20 (MeV·cm^2)/mg. The LTMR LET_{th} is lower than 20 because it still had multiple events at 20 (MeV·cm^2)/mg, and the LET_{th} point should be about 3 levels of magnitude below the highest SEE cross section value. In Fig 14, there was no data collected for LTMR at an LET of 20 (MeV·cm^2)/mg, so we do not know how different the LTMR would be from the DTMR at 1MHz.

On Fig 13 at 50 MHz, at every LET value tested, the LTMR data – except for the controls – had cross sections around an order of magnitude greater than the cross section from the DTMR data. These values show that there is indeed a significant difference between LTMR cross sections and DTMR cross section. This agrees with the initial hypothesis that the DTMR mitigation technique would indeed perform better than the LTMR mitigation technique.

B. Number of Inverters vs. Number of Buffers for LTMR and DTMR on the Shift Registers

The following charts show the cross section vs. linear energy transfer for LTMR and DTMR data separately at 50 MHz.

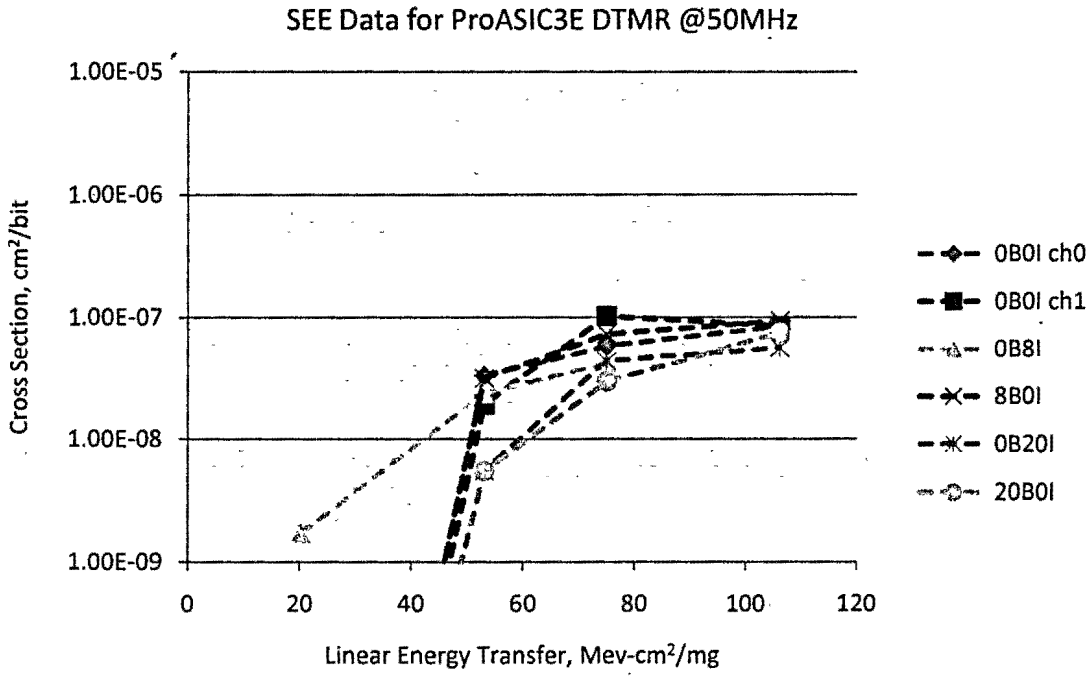

Figure 15. The cross section of shift registers with DTMR at 50MHz.

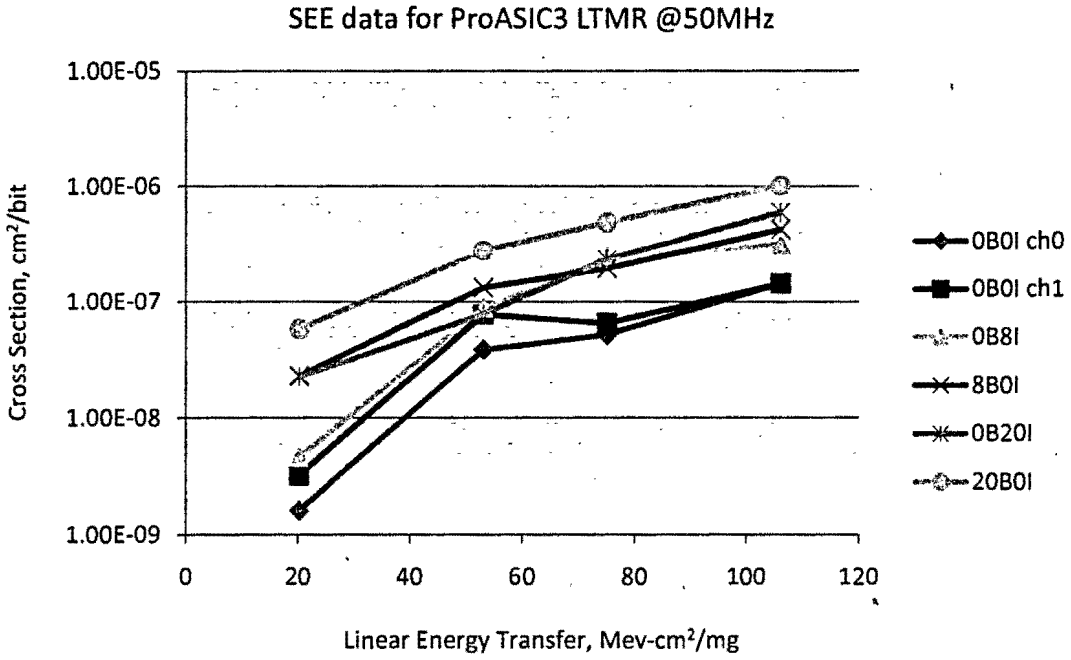

Figure 16. The cross section of shift registers with LTMR at 50MHz.

The first thing to notice is that there is no significant difference between inverters and buffers. Both Fig. 15 and Fig. 16 show channels 0 and 1 are almost exact, channels 2 and 3 are very similar, and channel 4 and 5 are also very similar. Since those channels have an equal number of inverters or buffers between the shift registers, and do not have a significant difference in cross sections, we can determine that either one can be used for testing purposing effectively on the ProASIC3E. When testing parts against each other, the same method should be used for both in case all FPGAs do not perform like this.

The effects of LTMR and DTMR on the cross sections based on the number of buffers or inverters can be determined as well. In the DTMR chart, Fig. 15, all the channels follow the same path, with close SEE cross sections. This follows the explanation of how DTMR works to mitigate single-event effects in combinatorial logic. Since DTMR triples the combinatorial logic, as well as the flip-flops, the errors from the combinatorial logic should be reduced as well, as the chart shows. The LTMR chart however, shows clear differences based on the number of inverters or buffers. When 20 buffers or inverters were used, there was about a 10x difference from the channels with no buffers or inverters. This is clear indication that LTMR does not reduce the effects of radiation events in the combinatorial logic supporting the initial thoughts on LTMR and DTMR.

C. Checkered vs. '0' pattern for LTMR and DTMR on the Shift Registers

The chart to the right shows the differences between the '0' pattern used and the checkered pattern used for LTMR at 100MHz.

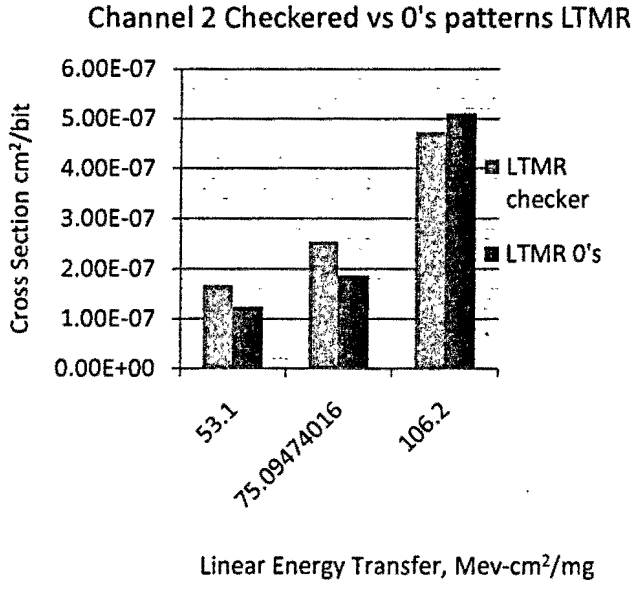

Figure 17. A bar graph showing the relationship between the checkered patter and the '0's pattern.

Figure 17 shows that while the checkered patter produces a slightly higher cross section, there is still not a significant difference between the patterns. This chart however, does not show all the data where the '0's pattern produced no errors during data runs while the checkered pattern was producing errors, especially using DTMR. This is an indication that there actually is a significant difference between the use of a '0's pattern and a checkered pattern. When performing SEE tests, a checkered pattern is more similar to space flight use than a '0's pattern is. Since it also produces more errors, it is a much better indication of what will actually happen in space.

D. LTMR vs. DTMR for counters and snapshots

The following charts show the cross sections vs. LET for the counter and snapshot test setup.

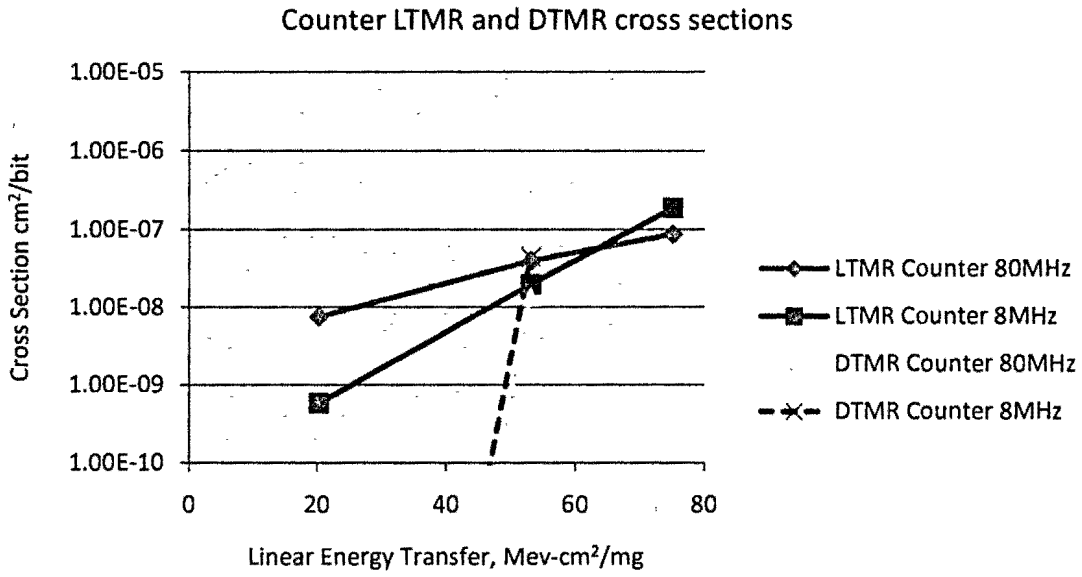

Figure 18. The cross sections for the counter. LTMR are solid lines, while DTMR are dashed lines.

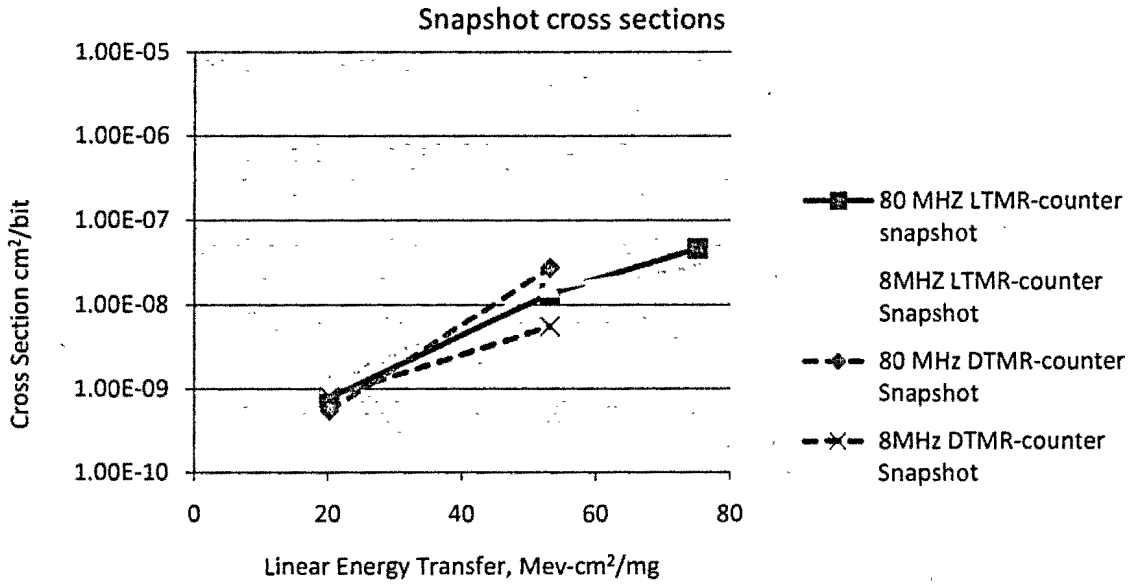

Figure 19. The cross sections for the snapshot in the counter setup. LTMR are solid lines, while DTMR are dashed lines.

The counter chart shows data very similar to that of the shift register. The LET_{th} of the LTMR technique is lower than that of the DTMR technique just like the shift register was. It is important to remember however, that the only LET values tested were 20, 53, 75, and 106 (MeV·cm^2)/mg. The actual LET_{th} of the DTMR will be between 20 and 53 (MeV·cm^2)/mg. There were no data gathered for the DTMR at a LET higher than 53 (MeV·cm^2)/mg because there was not enough time at the test facility.

Every snapshot was DTMR'd to provide the best mitigation. Since this test was primarily to examine the counter data, the snapshot errors were reduced as much as possible. The snapshot SEE cross sections were also not significantly different from the counters. This hints that differences between the counters and shift registers are insignificant, however the rest of the shift register data must also be examined.

E. Counters vs. Shift Registers

The following chart has plots for the counter cross sections as well as some of the shift register cross sections.

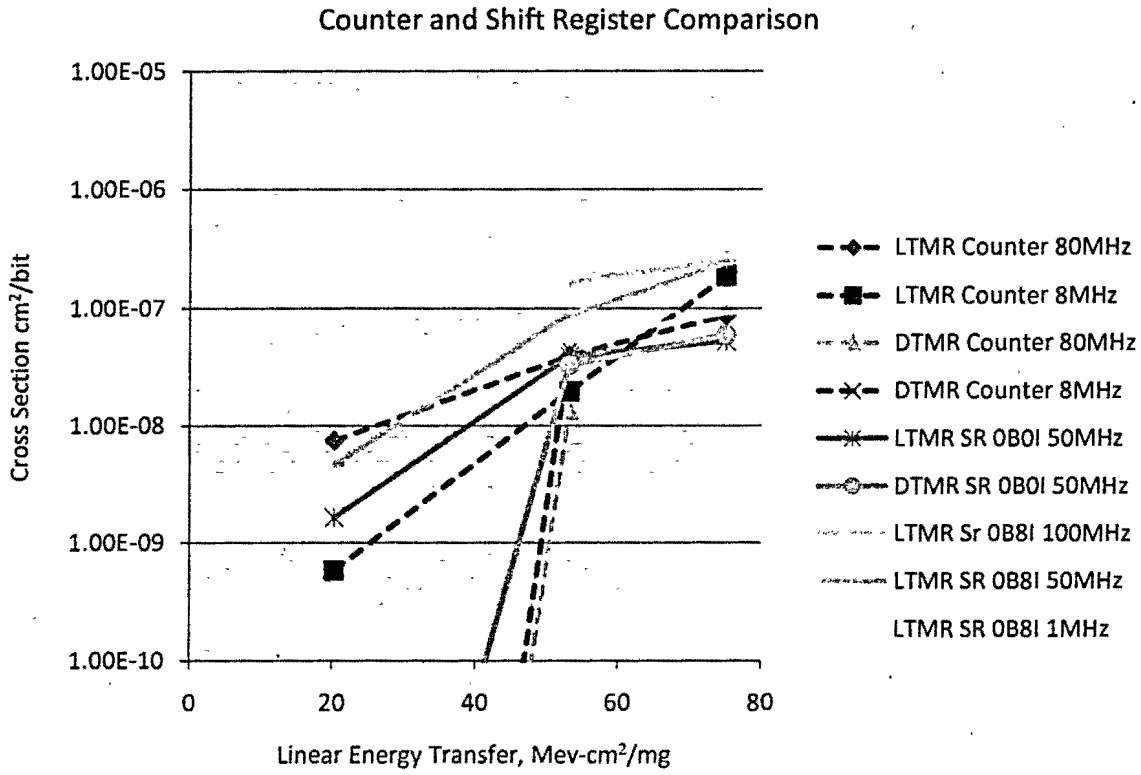

Figure 20. Cross sections for both the counter and some of the shift register channels. Counters are dashed lines while shift registers are solid lines.

Since the counters were not run at the same clock frequencies as the shift registers, the comparisons become more qualitative. As I explained about Fig. 18, we can see the similarities between the counters and shift registers with regards to LTMR and DTMR. The DTMR counters follow the same pattern which the DTMR shift registers did, with the higher LET_{th}. The 8 MHz LTMR counter is significantly lower than the 50 MHz register with 8 inverters at the low LET values, however, the 80 MHz LTMR counter is very close to that same shift register. Since the 80 MHz clock is closer in frequency to the 50 MHz shift register, it is a more reliable comparison than the 8 MHz counter. Along with this fact, and that at higher LET values there is no significant difference between the

counter and the shift register, the two methods of testing perform very similarly, with no major differences between them.

V. Conclusion

The results of this experiment show that the DTMR architecture mitigates single-event effect errors better than the LTMR architecture. The DTMR architecture also minimized the differences between registers with different numbers inverters or buffers between flip-flops, which tells us that the errors were probably caused by clock or enable faults, rather than logic faults. The LTMR data showed significantly higher cross sections for shift registers with higher numbers of inverters and buffers. This shows that LTMR does not reduce the errors caused by heavy ion strikes to the combinatorial logic. However, there was not a significant difference between the channels implemented with buffers and channels implemented with inverters. They both performed roughly the same during irradiation.

It was also visible that difference between using shift registers or counters was minimal. Implementing either one for the testing architecture would be a viable option for testing future FPGAs. One should consider the space application for the FPGA as well, and choose the architecture that most closely resembles it. However, the difference between the use of a checkered pattern and a '0's patter is significant. The '0's pattern reduced several of the runs to have no errors, while the checkered pattern was still producing errors. This is important to keep in mind for future tests, because a worst case architecture is desired for testing. The space application of an FPGA will be much more complex than the testing architecture and therefore more susceptible to SEE, so it is critical that the test architecture be as complex as possible.

Further testing is required to obtain more data on how the ProASIC3E responds to SEE radiation. The most important test data we need are varying levels of LET. A solid LET_{th} must be determined along with the $LET_{saturation}$ point where the cross section will no longer rise as LET continues to increase. Once those results are obtained, there can be a clear determination on how effective DTMR mitigates SEE over LTMR. It is also essential to compare these results with radiation hardened parts. Then a cost-effective decision will need to be made to determine the most efficient parts and method.

Appendix

COTS- Commercial off the shelf
DFF- D-Flip-Flop
DTMR- Distributed Triple Modular Redundancy
DUT- Device Under Test
FPGA- Field Programmable Gate Array
GTMR- Global Triple Modular Redundancy
LET- Linear Energy Transfer
LTMR- Localized Triple Modular Redundancy
TAMU- Texas A&M University
TMR- Triple Modular Redundancy
SEB- Single Event Burnout
SEE- Single Event Effect
SEFI- Single Event Functional Interrupt
SEGR- Single Event Gate Rupture
SEL- Single Event Latchup
SET- Single Event Transient
SEU- Single Event Upset

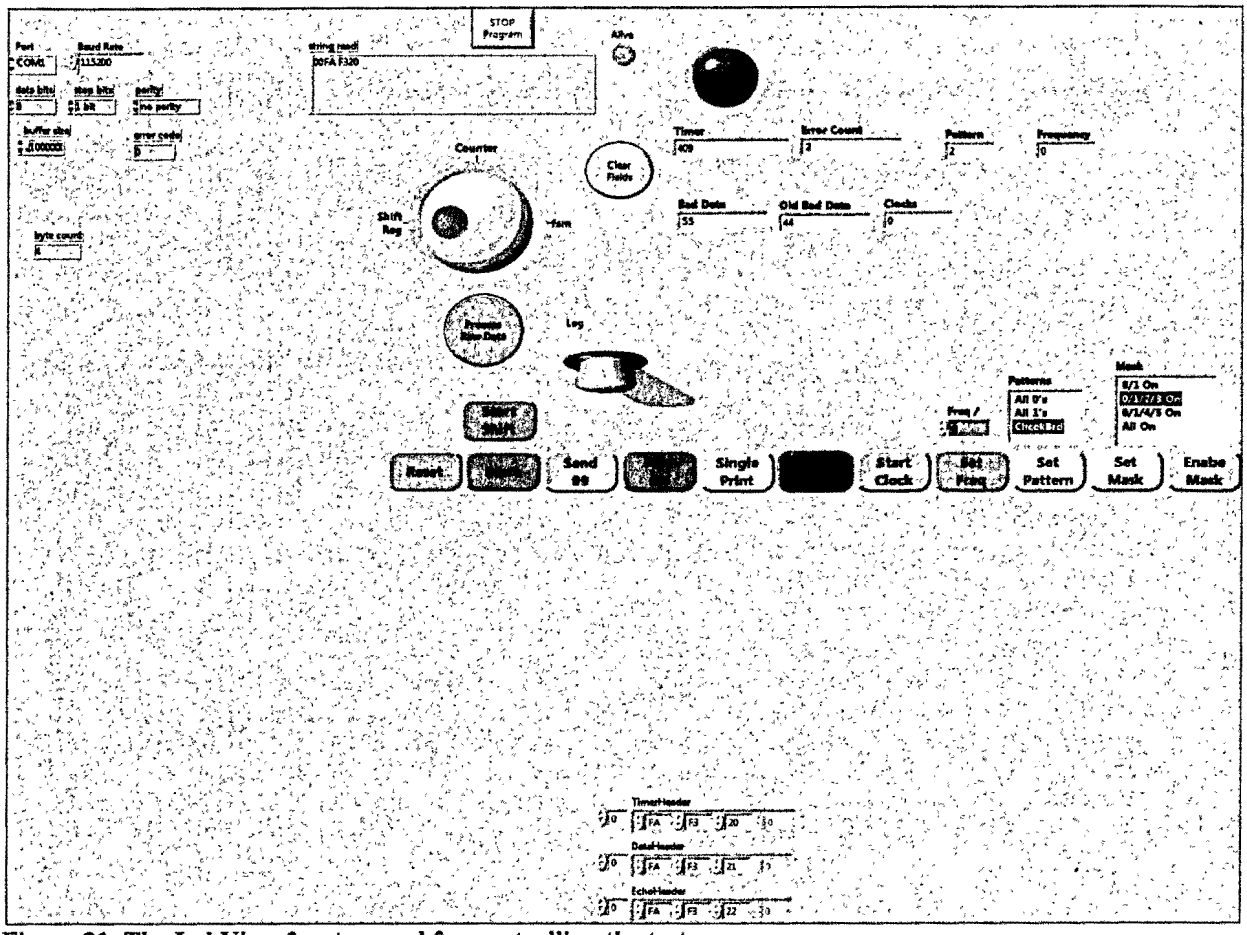

Figure 21. The LabView front pannel for controlling the test.

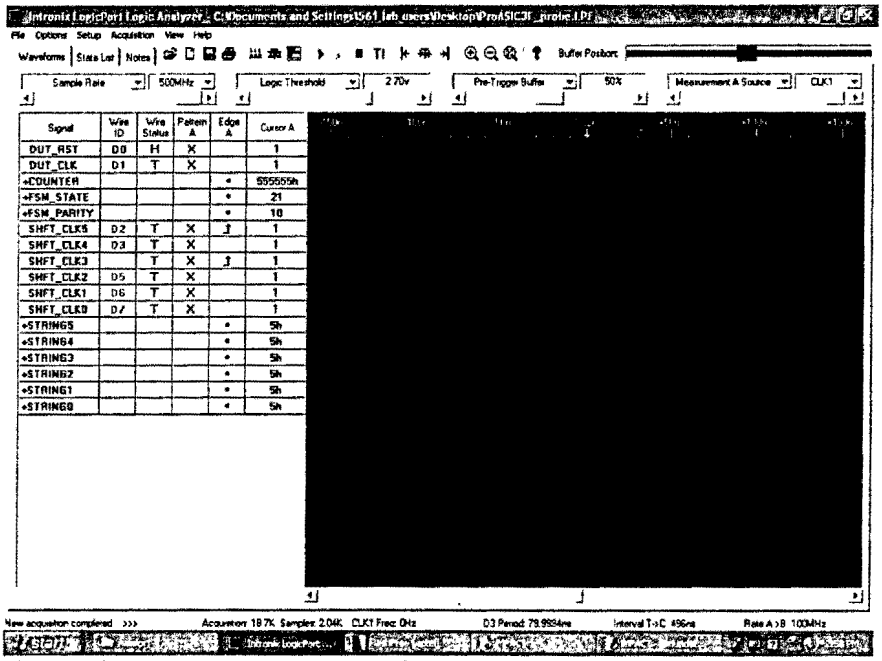

Figure 22. The output waveforms from the testing board and the DUT. The bottom 6 strings are the 6 channels in checkerboard pattern.

Acknowledgments

I would like to thank the entire Radiation Effects and Anaylses Group at NASA GSFC for their help and support on this project. I would especially like to thank Melanie Burg in assisting with this so much, along with my mentor Jonathan Pellish.

References

[1] Petersen, E., "Single-Event Analysis and Prediction," *IEEE Nuclear and Space Radiation Effects Conference Short Course,* Session III, IEEE, Snowmass, CO 1997, p III-5

[2] Barth, J. "Modeling Space Radiation Environments," *IEEE NSREC Short Course,* Session I, IEEE, Snowmass, CO 1997, pp I-35-38

[3] Berg, M. "Trading ASIC and FPGA considerations for System Insertion" *IEEE NSREC Short Course,* Session V, IEEE, Quebec City, Canada July 2009, p V-16

[4] Actel Datasheet: "ProASIC3E Flash Family FPGAs" http://www.actel.com/documents/PA3E_DS.pdf, V9, Aug. 2009

[5] Habinc, S., "Functional Triple Modular Redundancy: VHDL Design Methodology for Redundancy in Combinatorial and Sequential Logic" Design and Assessment Report, Gaisler Research, Dec. 2002, pp 4

[6] M.Berg "A Comprehensive Methodology for complex Field Programmable Gate Array Single Event Effects Test and Evaluation" IEEE Trans. Nucl. Sci. NS-56, 2009